```
 1: function main()
 2: Imax = 100;  % Number of Monte Carlo Runs
 3: % T. Cehelnik
 4:
 5: t = 0:0.01:5; % Setup time samples
 6:
 7: [D1t,V1t,S1t]=get_it(t,1);  %% Get Distance, Velocity, Signal training
 8: [D2t,V2t,S2t]=get_it(t,2);
 9: [D3t,V3t,S3t]=get_it(t,3);
10:
11: for I=1:Imax
12:
13: [D1,V1,S1]=get_it(t,1); % get D1
14:
15: [D2,V2,S2]=get_it(t,2);
16:
17: [D3,V3,S3]=get_it(t,3);
18:
19: C11 = corrcoef(S1t,S1);
20: C12 = corrcoef(S1t,S2);
21: C13 = corrcoef(S1t,S3);
22:
23: c11(I) = C11(1,2);
24: c12(I) = C12(1,2);
25: c13(I) = C13(1,2);
26:
27: C21 = corrcoef(S2t,S1);
28: C22 = corrcoef(S2t,S2);
29: C23 = corrcoef(S2t,S3);
30:
31: c21(I) = C21(1,2);
32: c22(I) = C22(1,2);
33: c23(I) = C23(1,2);
34:
35: C31 = corrcoef(S3t,S1);
36: C32 = corrcoef(S3t,S2);
37: C33 = corrcoef(S3t,S3);
38:
39: c31(I) = C31(1,2);
40: c32(I) = C32(1,2);
41: c33(I) = C33(1,2);
42: end
43:
44: figure(1)
45: plot(1:Imax,c11,'r',1:Imax,mean(c11),'r')
46: hold
```

```
47: plot(1:Imax,c12,1:Imax,mean(c12))
48: plot(1:Imax,c13,'k',1:Imax,mean(c13),'k')
49:
50: figure(2)
51: plot(1:Imax,c21,'r',1:Imax,mean(c21),'r')
52: hold
53: plot(1:Imax,c22,1:Imax,mean(c22))
54: plot(1:Imax,c23,'k',1:Imax,mean(c23),'k')
55:
56: figure(3)
57: plot(1:Imax,c31,'r',1:Imax,mean(c31),'r')
58: hold
59: plot(1:Imax,c32,1:Imax,mean(c32))
60: plot(1:Imax,c33,'k',1:Imax,mean(c33),'k')
61:
62:
63: figure(10)
64: plot(t,D1(:,1),'b',t,D1(:,2),'b.')
65: hold
66: plot(t,D1(:,3),'k',t,D1(:,4),'k.')
67: plot(t,D1(:,5),'m',t,D1(:,6),'m.')
68:
69: return
70:
71: function [D,V,S]=get_it(t,opt)
72: %% Get vector of point used to make motional command
73:
74: if(opt==1)
75: [rx,ry,rz]= sweepitx(t);
76: end
77:
78: if(opt==2)
79:     [rx,ry,rz]= sweepity(t);
80: end
81:
82: if(opt==3)
83:        [rx,ry,rz]= sweepitz(t);
84: end
85:
86:
87: %% add noise
88:
89: variance = 1.5;
90:
91: rx = rx + variance.*randn(1,1);
92: ry = ry + variance.*randn(1,1);
```

```
 93: rz = rz + variance.*randn(1,1);
 94:
 95: [r1,r2,r3,r4,r5,r6]= vectors(rx,ry,rz);
 96: d1 = dist(r1);d2 =dist(r2);d3 =dist(r4);d4 =dist(r4);
 97: d5 =dist(r5);d6 = dist(r6);
 98:
 99: % Get Distance D, Velocity V, and Voltage Signal S.
100: D = [dist(r1),dist(r2),dist(r3),dist(r4),dist(r5),...
101:         dist(r6)];
102: V = [vel(dist(r1)),vel(dist(r2)),vel(dist(r3)),...
103:         vel(dist(r4)),vel(dist(r5)),vel(dist(r6))];
104: S = [signal(d1),signal(d2),signal(d3),signal(d4),...
105:         signal(d5),signal(d6)]; % Simulate Signal
106: S = S./max(max(S)); %% Normalize Signal
107:
108: return
109:
110:
111: function [s] = signal(d)
112: %% Make signal proportional to 1/d;
113: s = 1./d;
114:
115: return
116:
117: function [v] = vel(d);
118: v = diff(d);
119:
120:
121: return
122:
123: function [d]= dist(r)
124:
125: d = sqrt(r.x.^2 +r.y.^2 +r.z.^2);
126: d = d';
127: return
128:
129: function [rx,ry,rz]=sweepity(t)
130:     zo = 10;
131:     xo =  0;
132:     yo = 0;
133:
134: %rx = xo + 10.*sin(-2*pi.*t);
135: %ry = yo;
136:
137: rx = xo ;
138: ry = yo + 10.*sin(-2*pi.*t); % vert
```

```
139:
140: rz = zo + 10.*cos(-2*pi.*t);
141:
142:     return
143:
144:  function [rx,ry,rz]=sweepitx(t)
145:     zo = 10;
146:     xo =  0;
147:     yo = 0;
148:
149: rx = xo + 10.*sin(-2*pi.*t); % horz
150: ry = yo;
151:
152: rz = zo + 10.*cos(-2*pi.*t);
153:
154:    return
155:
156: function [rx,ry,rz]=sweepitz(t)
157:     zo = 10;
158:     xo =  0;
159:     yo = 0;
160:
161: ry = yo + 10.*sin(-2*pi.*t);
162: rx = xo + 10.*cos(-2*pi.*t);;
163:
164: rz = zo ;
165:
166:     return
167:
168:
169:
170: %% vector r1 =
171: function [r1,r2,r3,r4,r5,r6]= vectors(rx,ry,rz);
172: x1 = 20;
173: y1 = 20;
174: z5 = 5;
175:
176: r1.x = rx - x1;
177: r2.x = rx + x1;
178: r3.x = rx;
179: r4.x = rx;
180: r5.x = rx;
181: r6.x = rx;
182:
183:
184: r1.y = ry;
```

```
185: r2.y = ry;
186: r3.y = ry- y1;
187: r4.y = ry+y1;
188: r5.y = ry-y1;
189: r6.y = ry+y1;
190:
191: r1.z = rz;
192: r2.z = rz;
193: r3.z = rz;
194: r4.z = rz;
195: r5.z = rz-z5;
196: r6.z = rz +z5;
197:
198: return
199:
```